| **Model** | **Figure** | **N** | **F** | **$\|\mathbf{u}_t\|$** | **g0 E dim** | **$\mathbf{u}_t$ E dim** | **C dim** | **G $L_2$** | **C $L_2$** | **KP** | **BS** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Monkey J Maze | Main 2,3 | 100 | 40 | 0 | 100 | - | - | 10 | - | 0.98 | 5 |
| Participant T5 Center-out | Main 3 | 64 | 20 | 3 | 64 | - | - | 250 | - | 0.95 | 5 |
| Monkey P Multi-session | Main 4 | 100 | 16 | 0 | 100 | - | - | 500 | - | 0.98 | 10 |
| Monkey P Single-session | Main 4 | 100 | 16 | 0 | 100 | - | - | 500 | - | 0.98 | 10 |
| Monkey J CursorJump | Main 5 | 128 | 50 | 4 | 150 | 100 | 128 | 25 | 25 | 0.98 | 10 |
| Monkey J Center-out | Main 6 | 128 | 50 | 4 | 150 | 100 | 128 | 25 | 25 | 0.98 | 2 |
| Participant T7 Center-out | Main 6 | 64 | 20 | 3 | 64 | 64 | 128 | 250 | 250 | 0.95 | 5 |
| Lorenz attractor | Supp. 2 | 64 | 3 | 0 | 64 | - | - | 250 | - | 0.95 | a.u. |
| Chaotic RNN | Supp. 3 | 200 | 20 | 0 | 200 | - | - | 2000 | - | 0.95 | a.u. |
| Input pulses | Supp. 6,7 | 200 | 20 | 1 | 200 | 128 | 128 | 2000 | 0 | 0.95 | a.u. |
| RNN Integrator | Supp. 8 | 200 | 20 | 1 | 128 | 128 | 128 | 2000 | 0 | 0.95 | a.u. |

**Supplementary Table 1**. Important hyper-parameters of LFADS models. Listed here are the most important LFADS parameters, relating primarily to model capacity. 'N' - number of units in the generator, 'F' - number of factors, $|\mathbf{u_t}|$ - number of inferred inputs, 'E' - encoder, 'C' - controller, 'G' - generator, 'KP' - keep probability in dropout layers, 'BS' - bin size (ms).

| Model | Figure | Electrode type | Signal post-processing |
|---|---|---|---|
| Monkey J Maze | Main 1, 2, 3 | Utah array | threshold crossings, spike sorted |
| Participant T5 Center-out | Main 3 | Utah array | threshold crossing |
| Monkey P Single-session | Main 4 | v-probe | threshold crossing |
| Monkey P Multi-session | Main 4 | v-probe | threshold crossing |
| Monkey J CursorJump | Main 5 | Utah array | threshold crossing |
| Monkey J Center-out | Main 6 | Utah array | threshold crossing |
| Participant T7 Center-out | Main 6 | Utah array | threshold crossing |

**Supplementary Table 2.** Signal collection technology and spike detection methods.

**Supplementary Note: Synthetic datasets**

Summary of synthetic datasets

We chose a variety of synthetic examples in an effort to show LFADS's ability to infer informative representations for dynamical systems of varying complexity. We ordered the synthetic examples roughly by complexity to build intuition. The examples are, in order,

1. The pendulum example (**Supp. Fig. 1**) - a cartoon (no actual data), simply intended to impart intuition using a well-known and tangible physical system.

2. The Lorenz model (**Supp. Fig. 2, Supp. Table 1**) - this simple model is now becoming standard in the field (e.g., [1,2]), as it is a simple and well-known example of a nonlinear, chaotic dynamical system, and easy to understand and visualize due to its 3D state space.

3. A synthetic RNN example with random connections and without input (**Supp. Fig. 3**) - this creates a much more complex high-dimensional dynamical system, intended to differentiate our method from common methods in the field that have difficulty modeling high-dimensional, highly nonlinear dynamics. This RNN does not have the same architecture as that used in LFADS.

4. A synthetic RNN example with simple pulse inputs (**Supp. Figs. 7,8**) - this provides a clear demonstration of the ability of LFADS to decompose an observed time series into both dynamics and inputs. This RNN does not have the same architecture as that used in LFADS.

5. A synthetic RNN trained to perform an integration-to-bound task, given a noisy 1-D input (**Supp. Fig. 9**). Integration-to-bound is a common model of decision-making in systems neuroscience. This example shows the utility of LFADS not only in modeling a network that is trained to perform a task, but also shows that LFADS can infer inputs in networks that are performing meaningful computations. This RNN does not have the same architecture as that used in LFADS.

Lorenz system

The Lorenz system is a set of nonlinear equations for three dynamic variables. Its limited dimensionality allows its entire state space to be visualized. The evolution of the system's state is governed as follows

$$\begin{aligned} \dot{y}_1 &= \sigma(y_2 - y_1) && (1) \\ \dot{y}_2 &= y_1(\rho - y_3) - y_2 && (2) \\ \dot{y}_3 &= y_1 y_2 - \beta y_3. && (3) \end{aligned}$$

We used the standard parameter values known for inducing chaos, $\sigma = 10$, $\rho = 28$, and $\beta = 8/3$, and used Euler integration with $\Delta t = 0.006$. As in [1], we simulated a population of neurons with firing rates given by linear read-outs of the Lorenz variables using random weights, followed by an exponential nonlinearity. Spikes from these firing rates were then generated by a Poisson process.

Our synthetic dataset consisted of 65 conditions, with 20 trials per condition. Each condition was obtained by starting the Lorenz system with a random initial state vector and running it for 1s.

Twenty different spike trains were then generated from the firing rates for each condition. Models were trained using 80% of the data (16 trials/condition) and evaluated using 20% of the data (4 trials/condition). While this simulation is structurally quite similar to the Lorenz system used in [1], we purposefully chose parameters that made the dataset more challenging. Specifically, relative to [1], we limited the number of observations to 30 simulated neurons instead of 50, decreased the baseline firing rate from 15 spikes/sec to 5 spikes/sec, and sped up the dynamics by a factor of 4.

## Chaotic RNNs as data generators

We tested the performance of each method at inferring the dynamics of a more complex nonlinear dynamical system, a fully recurrent nonlinear neural network with strong coupling between the units. We generated a synthetic dataset from an $N$-dimensional continuous time nonlinear, so-called, “vanilla" RNN,

$$\tau\, \dot{\mathbf{y}}(t) \quad = -\mathbf{y}(t) + \gamma\, \mathbf{W}^{y} \tanh(\mathbf{y}(t)) + \mathbf{B}\, \mathbf{q}(t). \; (40)$$

This makes a compelling synthetic case study for our method because many recent studies of neuroscientific data have used vanilla RNNs as their modeling tool (e.g. [3–7]). It should be stressed that the vanilla RNN used as the data RNN here does not have the same functional form as the network generator used in the LFADS framework, which is a GRU (see section *1.7*), although both have continuous variables and are not spiking models. For experiments in **Supp. Fig. 3**, we set $\mathbf{B} = \mathbf{q} = 0$, but we included an input for experiments in **Supp. Fig. 6**.

The elements of the matrix $\mathbf{W}^{y}$ were drawn independently from a normal distribution with zero mean and variance $1/N$ . We set $\gamma$ to either 1.5 or 2.5, both of which produce chaotic dynamics at a relatively slow timescale compared to $\tau$ (see [3] for more details). The smaller $\gamma$ value produces “gentler" chaotic activity in the data RNN than the larger value. Specifically, we set $N = 50$, $\tau = 0.025$ s and used Euler integration with $\Delta t = 0.01$ s. Spikes were generated by a Poisson process with firing rates obtained by scaling each element of $\tanh(\mathbf{y}(t))$ to take values in $[0,1]$, and then used as the rate in a Poisson process to give rates lying between 0 and 30 spikes/s.

Our dataset consisted of 400 conditions obtained by starting the data RNN at different initial states with elements drawn from a normal distribution with zero mean and unit variance. Firing rates were then generated by running the data RNN for 1 s, and 10 spiking trials were produced for each condition, yielding a total of 4,000 spiking trials. Models were trained using 80% of the data (8 trials/condition) and evaluated using 20% of the data (2 trials/condition).

## Inferring pulse inputs to a chaotic RNN

We tested the ability of LFADS to infer the input to a chaotic RNN (**Supp. Figs. 6,7**). In general, the problem of disambiguating dynamics from inputs is ill-posed, so we encouraged the dynamics to be as simple as possible by including an $L_2$ regularizer in the LFADS network generator (see **Supplementary Table 1**). We note that weight regularization is a standard technique that is nearly universally applied to neural network architectures.

Focusing on **Supp. Fig 6**, we studied the synthetic example of inferring the timing of a delta pulse input to a randomly initialized RNN. To introduce an input into the data RNN, the elements of **B** were drawn independently from a normal distribution with zero mean and unit variance. During each trial, we perturbed the network by delivering a delta pulse of magnitude 50, $q(t) = 50\delta(t - t_{pulse})$, at a random time $t_{pulse}$ between 0.25s and 0.75s (the full trial length was 1s). This pulse affects the underlying rates produced by the data RNN, which modulates the spike generation process. To test the ability of the LFADS model to infer the timing of these input pulses, we included in the LFADS model an inferred input with dimensionality of 1. We explored the same two values of $\gamma$ as in the synthetic example to model chaotic RNN dynamics, 1.5 and 2.5. Other than adding the input pulses, the data for input-pulse perturbations were generated as in the first data RNN example described above.

After training, which successfully inferred the firing rates, we extracted inferred inputs from the LFADS model (eqn. 15) by running the system 512 times for each trial, and averaging, defining $\overline{\mathbf{u}}_t = \langle \mathbf{u}_t \rangle_{\mathbf{g}_0, \mathbf{u}_{1:T}}$. To see how the timing of the inferred input was related to the timing of the actual input pulse, we determined the time at which $\overline{\mathbf{u}}_t$ reached its maximum value.

<u>Inferring white noise input in an RNN trained to integrate to bound</u>

We tested the ability of LFADS to infer the input to a vanilla RNN trained to integrate a noisy signal to a $+1$ or $-1$ bound. Weight matrices for this "data simulation RNN" were drawn independently from a Gaussian distribution with zero mean and variance $0.64/N$, and $L_2$ regularization was used during training. The noisy input signal was drawn from a Gaussian distribution with zero mean and variance $0.0625$. 800 conditions were generated with white noise inputs, and 5 spiking trials were generated per condition. This resulted in 4,000 1s spiking trials. 3,200 trials were used for training and 800 trials were used for validation.

After training LFADS on the integrate-to-bound data (simulated as above), inferred inputs ($\overline{\mathbf{u}}_t$) for a given trial were extracted by taking 1024 samples from the ($\overline{\mathbf{u}}_t$) posterior distribution produced by LFADS, and then averaging. These inferred inputs were then compared (using $R^2$) with the real inputs to the integrate-to-bound model, which were saved down previously during training.

## Supplementary Note: LFADS-related work in machine learning literature

Recurrent neural networks have been used extensively to model neuroscientific data (e.g. [3–7]), but the networks in these studies were all trained in a deterministic setting. An important recent development in deep learning has been the advent of the variational auto-encoder [8,9], which combines a probabilistic framework with the power and ease of optimization of deep learning methods. VAEs have since been generalized to the recurrent setting, for example with variational recurrent networks[10], deep Kalman filters[11], and the RNN DRAW network[12].

There is also a line of research applying probabilistic sequential graphical models to neural data. Recent examples include PLDS[13], switching LDS[14], GCLDS[15], and PfLDS[16]. These models employ a linear Gaussian dynamical system state model with a generalized linear model (GLM) for the emissions distribution, typically using a Poisson process. In the case of the switching LDS, the generator includes a discrete variable that allows the model to switch between linear dynamics. GCLDS employs a generalized count distribution for the emissions distribution. Finally, in the case of PfLDS, a nonlinear feed-forward function (neural network) is inserted between the LDS and the GLM.

Gaussian process models have also been explored. GPFA[17] uses Gaussian processes (GPs) to infer a time constant with which to smooth neural data and has seen widespread use in experimental laboratories. More recently, the authors of [1] have used a variational approach (vLGP) to learn a GP that then passes through a nonlinear feed-forward function to extract the single-trial dynamics underlying neural spiking data.

Additional work applying variational auto-encoding ideas to recurrent networks can be found in [18]. The authors of [11] have defined a very general nonlinear variational sequential model, which they call the Deep Kalman Filter (DKF). The authors of [19] applied recurrent variational architectures to problems of control from raw images. Finally, [20] applied dynamical variational ideas to sequences of images. Due to the generality of the equations in many of these references, LFADS is likely one of many possible instantiations of a variational recurrent network applied to neural data (in the same sense that a convolutional network architecture applied to images is also a feed-forward network, for example).

The LFADS model decomposes the latent code into an initial condition and a set of innovation-like inferred inputs that are then combined via an RNN to generate dynamics that explain the observed data. Recasting our work in the language of Kalman filters, our nonlinear generator is analogous to the linear state estimator in a Kalman filter, and we can loosely think of the inferred inputs in LFADS as innovations in the Kalman filter language. However, an “LFADS innovation” is not strictly defined as an error between the measurement and the read-out of the state estimate. Rather, the LFADS innovation may depend on the observed data and the generation process in extremely complex ways.

## Supplementary References

1. Zhao, Y. & Park, I. M. Variational Latent Gaussian Process for Recovering Single-Trial Dynamics from Population Spike Trains. *Neural Comput.* **29,** 1293–1316 (2017).
2. Linderman, S. *et al.* Bayesian Learning and Inference in Recurrent Switching Linear Dynamical Systems. *Artificial Intelligence and Statistics* 914–922 (2017). at <http://proceedings.mlr.press/v54/linderman17a.html>
3. Sussillo, D. & Abbott, L. F. Generating coherent patterns of activity from chaotic neural networks. *Neuron* **63,** 544–557 (2009).
4. Mante, V., Sussillo, D., Shenoy, K. V & Newsome, W. T. Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature* **503,** 78–84 (2013).
5. Carnevale, F., de Lafuente, V., Romo, R., Barak, O. & Parga, N. Dynamic Control of Response Criterion in Premotor Cortex during Perceptual Detection under Temporal Uncertainty. *Neuron* **86,** 1067–1077 (2015).
6. Sussillo, D., Churchland, M. M., Kaufman, M. T. & Shenoy, K. V. A neural network that finds a naturalistic solution for the production of muscle activity. *Nat. Neurosci.* **18,** 1025–1033 (2015).
7. Rajan, K., Harvey, C. D. & Tank, D. W. Recurrent Network Models of Sequence Generation and Memory. *Neuron* **90,** 1–15 (2016).
8. Kingma, D. P. & Welling, M. Auto-Encoding Variational Bayes. *arXiv [stat.ML]* (2013). at <http://arxiv.org/abs/1312.6114v10>
9. Rezende, D. J., Mohamed, S. & Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. in *International Conference on Machine Learning, 2014* (2014).
10. Chung, J. *et al.* A Recurrent Latent Variable Model for Sequential Data. in *Advances in Neural Information Processing Systems (NIPS)* (2015).
11. Krishnan, R. G., Shalit, U. & Sontag, D. Deep Kalman Filters. *arXiv Prepr. arXiv1511.05121* (2015).
12. Gregor, K., Danihelka, I., Graves, A., Rezende, D. J. & Wierstra, D. DRAW: A Recurrent Neural Network For Image Generation. *arXiv [cs.CV]* (2015). at <http://arxiv.org/abs/1502.04623>
13. Macke, J. H. *et al.* Empirical models of spiking in neural populations. *Advances in neural information processing systems* 1350–1358 (2011).
14. Petreska, B. *et al.* in *Advances in Neural Information Processing Systems 24* (eds. Shawe-Taylor, J., Zemel, R. S., Bartlett, P. L., Pereira, F. & Weinberger, K. Q.) 756–764 (Curran Associates, Inc., 2011). at <http://papers.nips.cc/paper/4257-dynamical-segmentation-of-single-trials-from-population-neural-data.pdf>
15. Gao, Y., Buesing, L., Shenoy, K. V. & Cunningham, J. P. High-dimensional neural spike train analysis with generalized count linear dynamical systems. *Adv. Neural Inf. Process. Syst.* 1–9 (2015). at <https://bitbucket.org/mackelab/pop_spike_dyn/downloads/Gao_Buesing_2015_GCLDS.

pdf>

16. Gao, Y., Archer, E. W., Paninski, L. & Cunningham, J. P. in *Advances in Neural Information Processing Systems 29* (eds. Lee, D. D., Sugiyama, M., Luxburg, U. V, Guyon, I. & Garnett, R.) 163–171 (Curran Associates, Inc., 2016). at <http://papers.nips.cc/paper/6430-linear-dynamical-neural-population-models-through-nonlinear-embeddings.pdf>

17. Yu, B. M. *et al.* Gaussian-Process Factor Analysis for Low-Dimensional Single-Trial Analysis of Neural Population Activity. *J. Neurophysiol.* **102,** 614–635 (2009).

18. Bayer, J. & Osendorfer, C. Learning stochastic recurrent networks. *arXiv Prepr. arXiv1411.7610* (2014).

19. Watter, M., Springenberg, J., Boedecker, J. & Riedmiller, M. Embed to control: A locally linear latent dynamics model for control from raw images. in *Advances in Neural Information Processing Systems* 2746–2754 (2015).

20. Karl, M., Soelch, M., Bayer, J. & van der Smagt, P. Deep variational Bayes filters: Unsupervised learning of state space models from raw data. *arXiv Prepr. arXiv1605.06432* (2016).